FYS-STK4155 - Project 2

Geir Tore Ulvik, Idun Klvstad

December 22, 2018

Abstract

In this work four methods to sample or manipulate input data to learning algorithms are explored: Random over-sampling, SMOTE, ADASYN, and balanced weighting. Several way to measure classifier performance are used, and the performance of logistic regression and a random forest classifier is evaluated based on how they perform on a binary classification problem. The data set chosen is payment data from an important bank in Taiwan, where the data describes credit card holders, and the goal is predicting whether a customer will default the next payment or not. The results presented show that logistic regression can classify the credit card data perfectly by applying a balanced weighting of the inputs. Random forests reached an accuracy of 95% (cross-validation score). A possible explanation for the success of weighting the inputs compared to the other sampling methods is that some features in the data set may be far more crucial in determining the class than others, and weighting is the most efficient way of emphasizing these features through the learning process. Random forests do not gain similar improvement with the same weighting, showing most improvement when using random over-sampling.

1 Introduction

In this project, we explore different ways to improve results of two binary classifiers without hyperparameter tuning - Logistic regression and Random forests. The methods used are of the over-sampling type and are as follows: Random over-sampling, SMOTE, ADASYN, and balanced weighting of inputs. The data set chosen is payment data from an important bank in Taiwan, where the data describes credit card holders, and the goal is predicting whether a customer will default the next payment or not. It is provided by UCI and has also been the subject of study by [1]: Credit Card Data Some of our figures are compared to those of the article as it is a nice reference point for logistic regression, but doesn't consider random forests (only classification trees).

The report assumes the reader is familiar with logistic regression and random forests. If not, a great introduction to both topics (and many others) have been written by Mehta et. al ([2]). A brief introduction to the sampling and manipulation methods, and methods used to measure classifier performance is included in the theory section.

Most of the analysis and models are implemented using existing frameworks:

- Scikit-Learn
- Scikit-Plot
- Imbalanced-Learn

The package 'Imbalanced-Learn' was used to perform the Random Over-Sampling, SMOTE, and ADASYN algorithms. Apart from the cumulative gain chart, plots were generated using the scikit-plot package.

Code, data and figures are available at the following GitHub address: GitHub repository

2 Theory

2.1 The Data Set

As mentioned in the introduction The data set chosen is payment data from an important bank in Taiwan, where the data describes credit card holders, and the goal is predicting whether a customer will default the next payment or not.

The data is from 2005. Among the total 25,000 observations, 5529 observations (22.12%) are the cardholders with default payment. [1]

The research of this data set employed a binary variable, default payment (Yes = 1, No = 0), as the response variable. The study reviewed the literature and used the following 23 variables as explanatory variables:

- X1: Amount of the given credit (NT dollar): it includes both the individual consumer credit and his/her family (supplementary) credit.
- X2: Gender (1 = male; 2 = female).
- X3: Education (1 = graduate school; 2 = university; 3 = high school; 4 = others).
- X4: Marital status (1 = married; 2 = single; 3 = others).
- X5: Age (year).
- X6 - X11: History of past payment. We tracked the past monthly payment records (from April to September, 2005) as follows: X6 = the repayment status in September, 2005; X7 = the repayment status in August, 2005; . . .;X11 = the repayment status in April, 2005. The measurement scale for the repayment status is: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; . . .; 8 = payment delay for eight months; 9 = payment delay for nine months and above.
- X12-X17: Amount of bill statement (NT dollar). X12 = amount of bill statement in September, 2005; X13 = amount of bill statement in August, 2005; . . .; X17 = amount of bill statement in April, 2005.
- X18-X23: Amount of previous payment (NT dollar). X18 = amount paid in September, 2005; X19 = amount paid in August, 2005; . . .;X23 = amount paid in April, 2005.

[?]

The aim of doing such a prediction is to to reduce damage and uncertainty that occur when a bank over-issues cash and credit cards to unqualified cardholders [1].

When such a crisis occur it is a challenge for both banks and cardholders, and a risk prediction is therefore a very useful tool.

2.2 Imbalanced Data in Classification

A common challenge in classification is imbalanced data, in which a large amount of the labeled data belongs to just one or a few of the classes. For binary classification, if 90% of the data belongs to one of the classes, then the classifier is likely to end up placing every single input in that class, as it will bring its accuracy to 90%. Techincally, this accuracy is correct, but it's not very useful since the decision isn't at all affected by the features of the input. Accuracy alone isn't a good enough measure of performance to reveal this.

Fortunately, since this is common, a number of methods have been developed to combat the issue, some of which are described below.

2.3 Resampling and Weighting

In resampling there are essentially two main categories: Under-sampling over-sampling. The difference between them is that over-sampling works with somehow generating more samples of the minority class, while under-sampling uses a reduced amount of samples from the majority class. Weighting the samples is a differente approach in which the samples labeled as the minority class are weighted higher than the others during training.

2.3.1 Naive Random Over-sampling

A very straightforward way to balance a dataset, is to choose random samples from the minority class, with replacement, until there is roughly equal amounts of samples belonging to each class.

2.3.2 SMOTE

SMOTE - Synthetic Minority Over-sampling Technique, as the name suggests will actually synthesize samples from the minority class in order to over-sample, instead of sampling with replacement. This is done by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors. Depending upon the amount of over-sampling required, neighbors from the k nearest neighbors are randomly chosen. The result of synthesizing rather than choosing with replacement is that the decision region is forced to become more general. See [3] for a more detailed explanation of the methods involved.

2.3.3 ADASYN

Like SMOTE, ADASYN (Adaptive Synthetic Sampling) generates synthetic samples in order to balance the number of samples in each class. The difference is mainly that ADASYN uses a density distribution as a criterion to automatically decide the number of synthetic samples for each sample in the minority class. The density distribution is a measurement of the distribution of weight for different minority class examples according to their level of difficulty in learning. This way, ADASYN effectively forces the learning algorithm to focus more on examples that are difficult to learn.

2.3.4 Balanced Weighting

Scikit-learn's logistic regressor comes with it's own form of handling of imbalanced data - weighting. It is a streightforward approach in which the values of targets are used to to automatically adjust weights inversely proportional to class frequencies in the input data as

$$\frac{samples}{classes \cdot np.bincount(targets)}$$

2.4 Assessing the Performance of Models

If classification accuracy is not enough to gauge whether a model is performing well, or well in the desired way, alternative way to measure performance must be explored. For cases of imbalanced data there are a few widely used methods that reveal information about the model that the simple accuracy metric can't.

2.4.1 Confusion Matrix

A confusion matrix is an n by n matrix containing correct classifications on the diagonal, and false positives and negatives in the off-diagonal elements. An example of such a matrix could be the following table: In the table above (1), the diagonal elements $i = j$

	True Cat	True Dog	True Rabbit
Predicted Cat	**5**	2	0
Predicted Dog	3	**3**	2
Predicted Rabbit	0	1	**11**

Table 1: Confusion matrix for an example classification where the classes are Cat, Dog and Rabbit. Correct classifications in bold.

are the correct classifications, while the other elements correspond to cases where the model predicted class i but should've predicted class j. The confusion matrix thus gives information about false positives and false negatives, in addition to classification accuracy. This is very useful in cases where for example false positives can be readily ignored or filtere later, but false negatives may have severe consequences. An example of this could be detection of cancer, in which a false positive can be ruled out from further testing, while a false negative may lead to a patient being sent home when actually needing help. For a more in-depth look at confusion matrices see [4].

2.4.2 Cumulative Gains Chart

Cumulative Gains Charts, often referred to as 'Lift Charts' (actually different ways to represent the same concept), can be used to gain a different insight. The chart lets us compare a binary classifier to both an ideal case and a 'random choice' case at the same time. The ideal classifier will predict an input's category with 100% confidence, so the probability will be 1.0 for the correct class and 0 for the wrong class. Sorting the predicted probabilities for the desired class (usually class 1) in descending order, will for the ideal case leave all members for class 1 on top, and plotting them in the order of appearance will give a steep curve. Random choice is used as a baseline for the chart, and plotting the model's curve should place it in between the random choice and ideal case. An example is shown in figure 1.

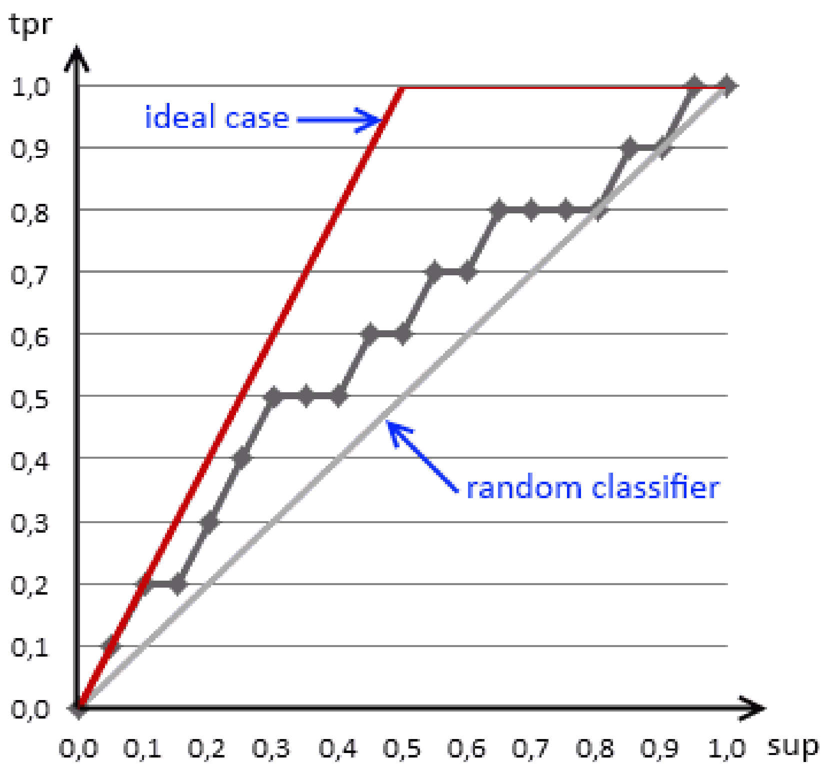


Figure 1: Example of a cumulative gain chart. Image borrowed from [5].

2.4.3 Receiver Operating Characteristic

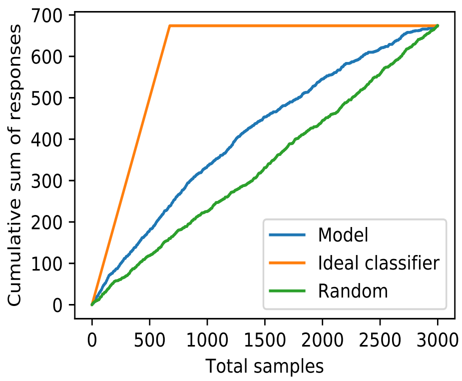
The Receiver Operating Characteristic (ROC) is a widely used measure of a classifiers performance . The performance is measured as the effect of the true positive rate (TPR) and the false positive rate (FPR) as a function of thresholding the positive class. To evaluate the ROC curve for a model, traditionally the Area Under the Curve (AUC) is used, which ranges from 0 (an ideal "opposite" classifier) to 1.0 (an ideal classifier) with 0.5 indicating a random choice classifier, [6](III.c). For a thorough explanation of ROC curves and the underlying concepts, see [4].

3 Results

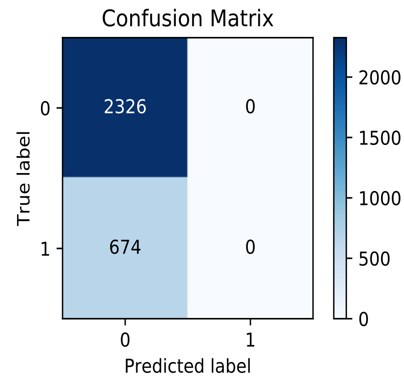
3.1 Logistic Regression

Results from the logistic regression are included in figures 2, 3, 4, 5. Figure 2 shows the performance of the logistic regressor without any added sampling techniques, and serves as a baseline for comparison. Due to observing varying mean accuracy for the different methods, cross-validation accuracies were also calculated, and are presented in table **??**. All the sampling methods were able to produce perfomances rivaling that of balanced weighting, but with less stability. The mean accuracy varied between 0.60 to 0.999, which is also reflected in the cross-validation scores.

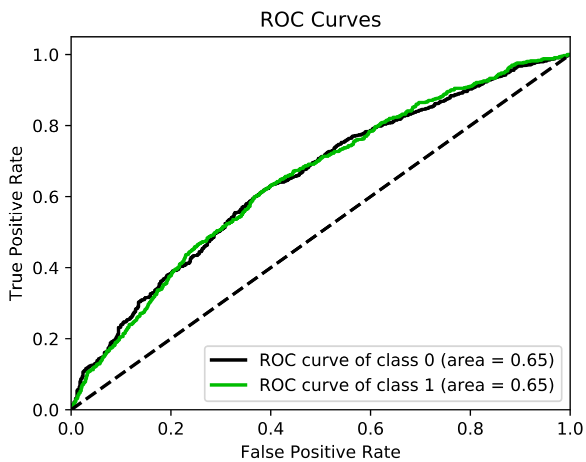
Figure 6 shows the analysis results for the random forest classifier. Based on the cross-validation accuracy scores included in table 2, only results obtained when applying random over-sampling were used for plotting. Note that the plots are generated from single runs of the code, not averaged over several.



(a)

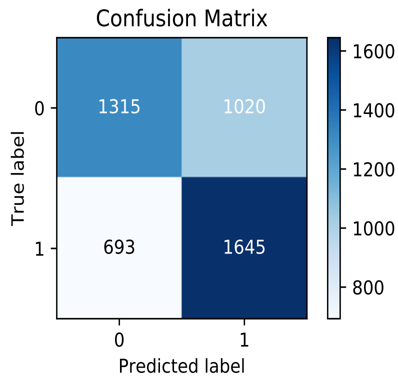


(b)

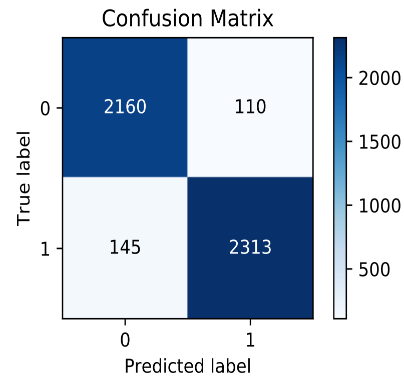


(c)

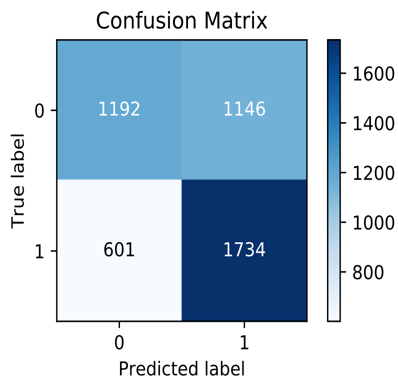
Figure 2: Performance analysis for basic logistic regression with no added sampling for balancing.



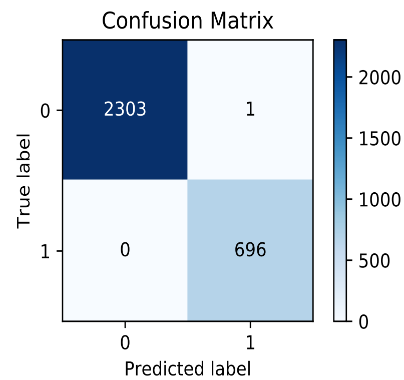
(a) Random oversampling



(b) ADASYN

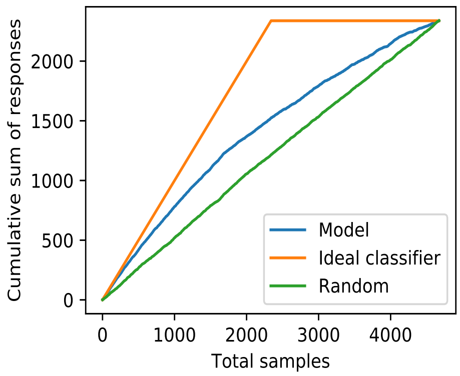


(c) SMOTE

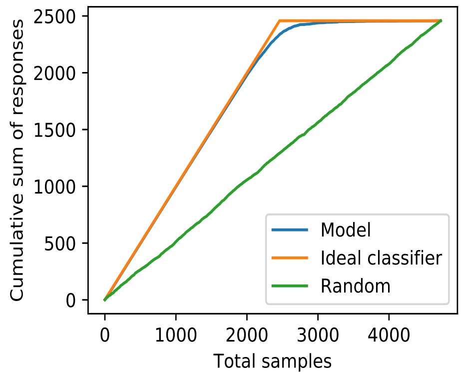


(d) Balanced weighting

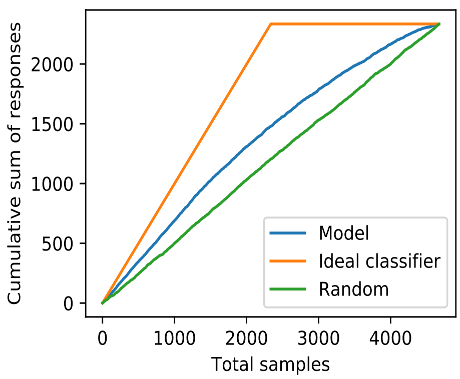
Figure 3: Confusion matrices for the logistic regression model using different resampling methods to balance the data set.



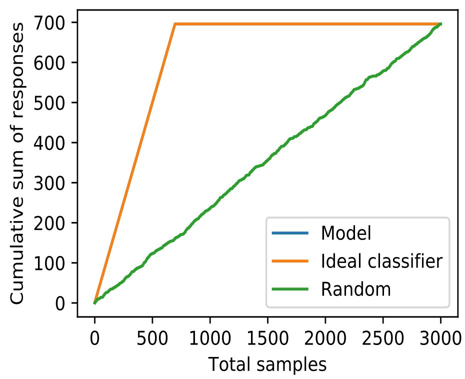
(a) Random oversampling



(b) ADASYN

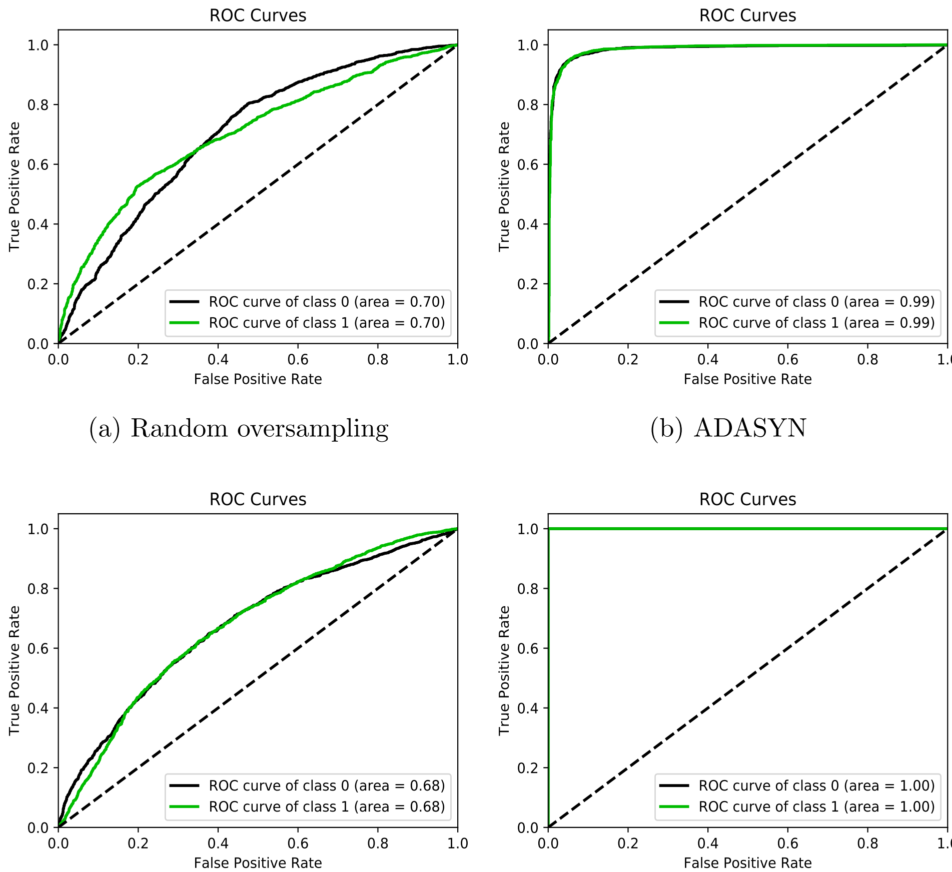


(c) SMOTE



(d) Balanced weighting

Figure 4: Cumulative gain chart for the logistic regression model using different resampling methods to balance the data set.



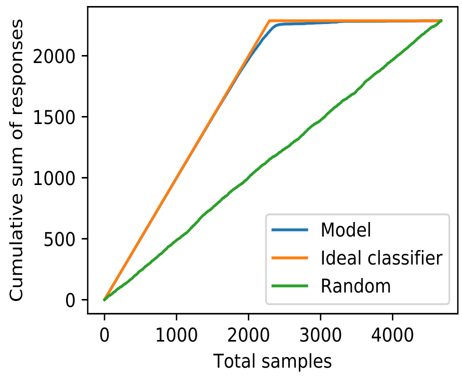
(a) Random oversampling

(b) ADASYN

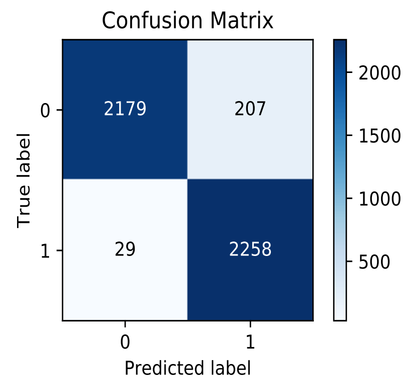
(c) SMOTE

(d) Balanced weighting

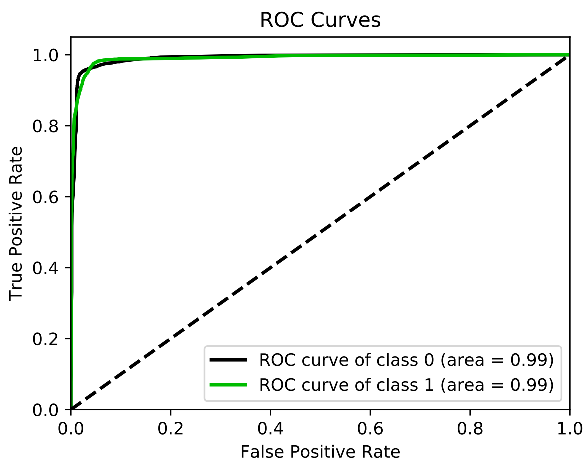
Figure 5: ROC curves for the logistic regression model using different resampling methods to balance the data set.



(a) Cumulative Gain Chart



(b) Confusion Matrix



(c) ROC

Figure 6: Performance analysis for random forest classification. Only the results obtained using random over-sampling are included as they provided the best cross-validation accuracy scores (table 2).

Table 2: Cross-validation accuracies for the different sampling methods used with both logistic regression and random forest classifier.

Sampling method	Logistic	Random Forest
None	0.78 ± 0.00	0.82 ± 0.02
Random oversampling	0.85 ± 0.35	0.94 ± 0.03
SMOTE	0.84 ± 0.29	0.84 ± 0.14
ADASYN	0.95 ± 0.12	0.82 ± 0.12
Balanced weighting	1.00 ± 0.00	0.81 ± 0.02

4 Discussion

For comparison of logistic regression, the figure 7 is borrowed from [1].

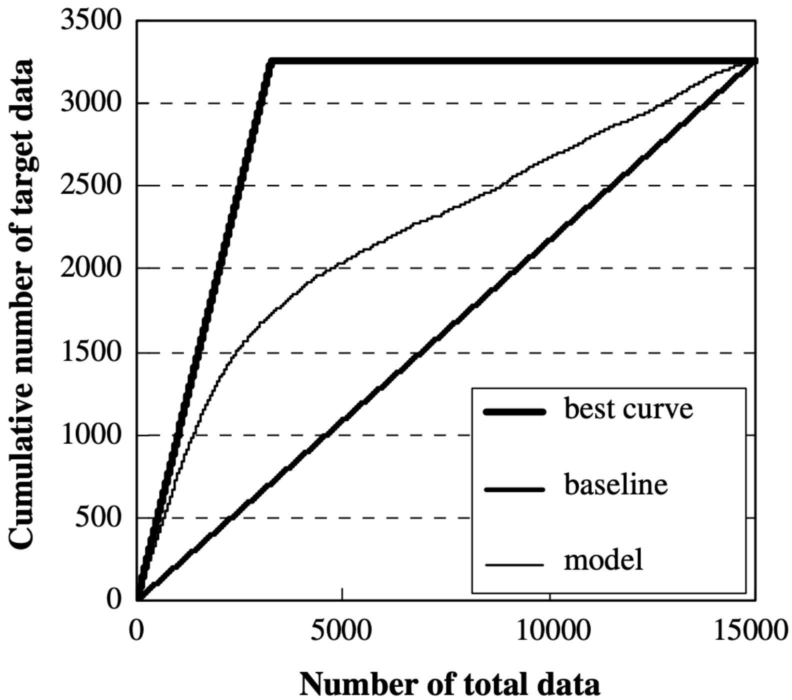


Fig. 3. Lift chart of logistic regression.

Figure 7: Cumulative gain char for logistic regression in [1] (Figure 3)

In [1], logistic regression performs slightly better than can be seen in figure 2. From the slightly steeper curve of in the cumulative gain chart (7). Logistic regression is, however, ill-suited to perform well on a dataset that is imbalanced like this one. As can be seen from figures 4, 3, 5, drastic improvement can be made. When the data is imbalanced, the logistic algorithm classifies every input into one class, which can be seen in the confusion matrix in figure 2(b). The sampling methods seem to remedy this, likely because the cost-function of the model can no longer be minimized by putting all inputs in the same class.

SMOTE reaches roughly the same performance as random over-sampling, but the ROC curves in figure 5(a, c) shows the methods lead to a slightly different curvature. This may be due to random-oversampling using extra copies of the existing inputs in the miniature class, while SMOTE synthesizes new samples that are only similar to existing ones.

ADASYN performs very well reaching a cross-validation accuracy of 95%, but is still outclassed by the built-in balanced weighting of scikit-learn's model, which performs perfectly, despite its straightforward nature. It would seem then, that even though synthesizing more data in the minority class does improve the performance, this type of problem responds extremely well to weighting.

Based on the superb performance of the model with weighting, a possible explanation for this could be that heavier weighting of the minority class is the most efficient way to expose which features of the inputs are key when classifying it. It would be interesting to see if this behaviour emerges for other datasets of a similar nature, when applying logistic regression.

In figure 6, Random Forests are applied to the same data set. Random forests could not reach the performance of logistic regression, but peaked with random over-sampling just around the level of the logistic model with ADASYN. Balanced weighting (table2) did not perform as well as for logistic regression. The other sampling methods resulted in very little improvement over no extra sampling. This may be due to lack of hyperparameter tuning, which was not a focus of this project.

5 Conclusion

In this work four methods to sample or manipulate input data to learning algorithms are explored: Random over-sampling, SMOTE, ADASYN, and balanced weighting. Using Cumulative Gain Charts, Confusion Matrices and ROC curves, and cross-validation, the performance of a logistic regression algorithm, and a random forest classifier is evaluated based on how they perform on a binary classification problem. The data set chosen is payment data from an important bank in Taiwan, where the data describes credit card holders, and the goal is predicting whether a customer will default the next payment or not. This allowed for comparisons with [1]. The results presented show that logistic regression can classify the credit card data perfectly by applying a balanced weighting of the inputs. Random forests reached an accuracy of 95% (cross-validation score). A possible explanation for the success of weighting the inputs compared to the other sampling methods is presented. Specifically, some features in the data set may be far more crucial in determining the class than others, and weighting is the most efficient way of emphasizing these features through the learning process. This works only for the logistic model, however. For random forests, the most improvement was found when using random over-sampling.

For future studies, an in-depth look at which features are the most important, and how the different models evaluate this, could be very interesting, as it may reveal some underlying effects of the sampling techniques.

References

[1] I-Cheng Yeh and Che hui Lien. The comparisons of data mining techniques for the predictive accuracy o f probability of default of credit card clients. *Expert Systems with Applications*, 36(2, Part 1):2473 – 2480, 2009.

[2] Pankaj Metha et al. A high-bias, low-variance introduction to machine learning for physicists. 03 2018.

[3] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *arXiv e-prints*, page arXiv:1106.1813, June 2011.

[4] Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861 – 874, 2006. ROC Analysis in Pattern Recognition.

[5] `http://mlwiki.org/index.php/Cumulative_Gain_Chart`.

[6] Robert Solli, John M. Aiken, Rachel Henderson, and Marcos D. Caballero. Examining the relationship between student performance and video interactions. *arXiv e-prints*, page arXiv:1807.01912, July 2018.